

202

"Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this. But in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow this ground. The brave men, living and dead who struggled here have consecrated it far above our poor power to add or detract. The world will little note nor long remember what we say here, but it can never forget what they did here. It is for us the living rather to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us--that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion--that we here highly resolve that these dead shall not have died in vain, that this nation under God shall have a new birth of freedom, and that government of the people, by the people, for the people shall not perish from the earth."

FIG. 2a

204

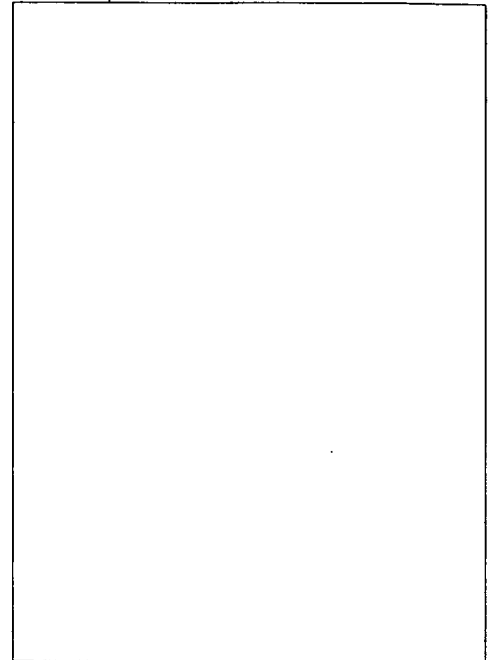


FIG. 2b

206

"Fourscore and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this. But in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow this ground. The brave men, living and dead who struggled here have consecrated it far above our poor power to add or detract."

208

The world will little note nor long remember what we say here, but it can never forget what they did here. It is for us the living rather to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us--that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion--that we here highly resolve that these dead shall not have died in vain, that this nation under God shall have a new birth of freedom, and that government of the people, by the people, for the people shall not perish from the earth."

FIG. 2c

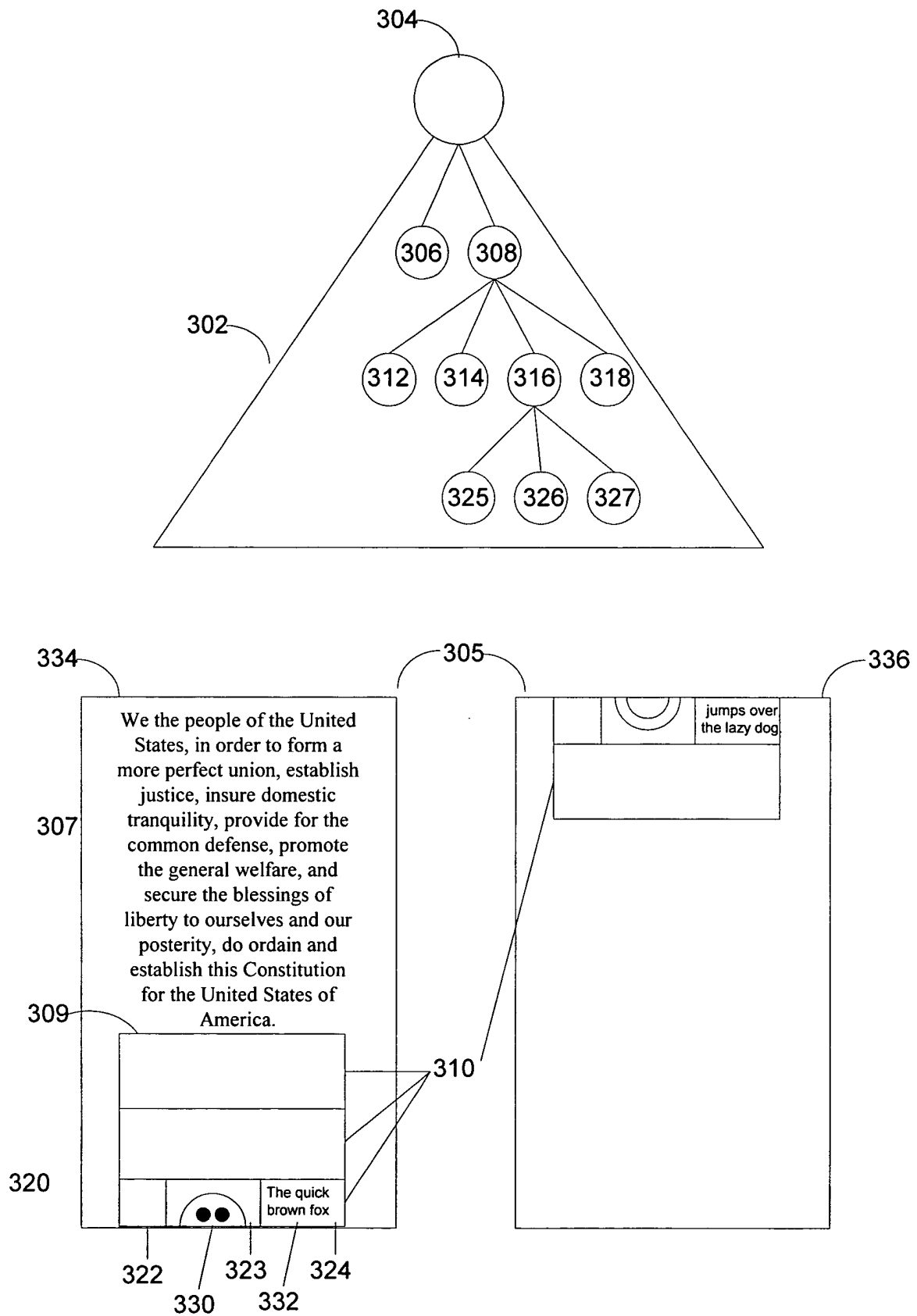


FIG. 3

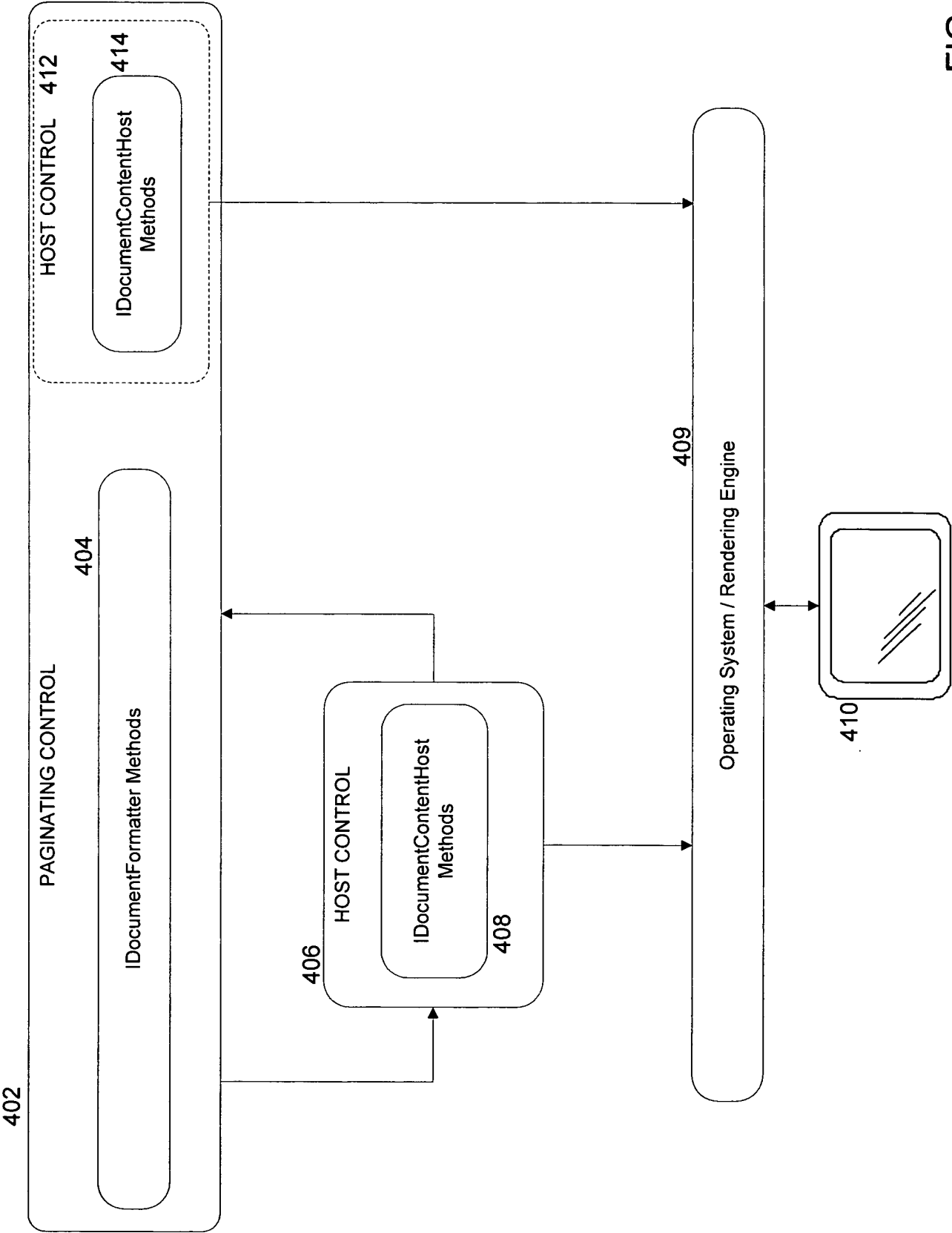
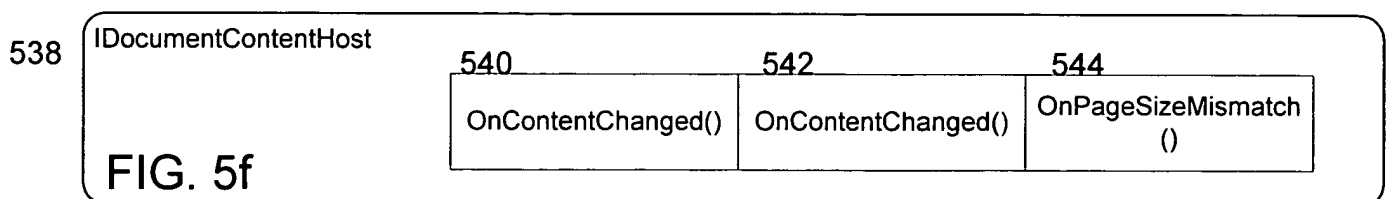
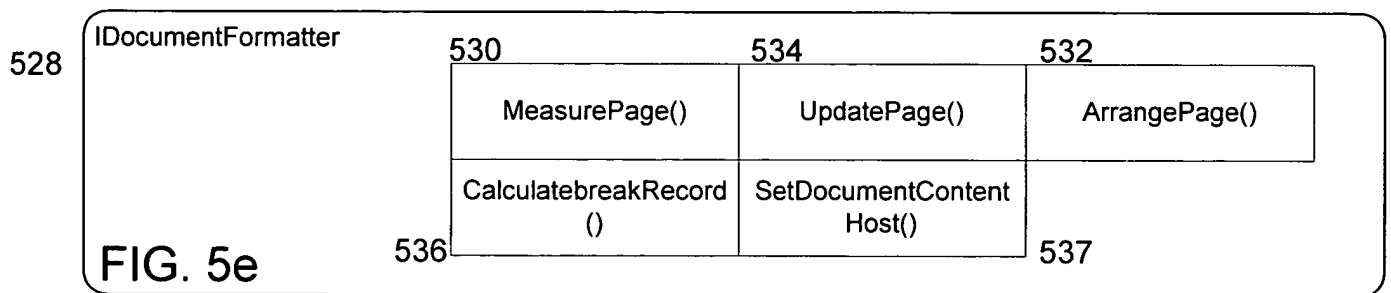
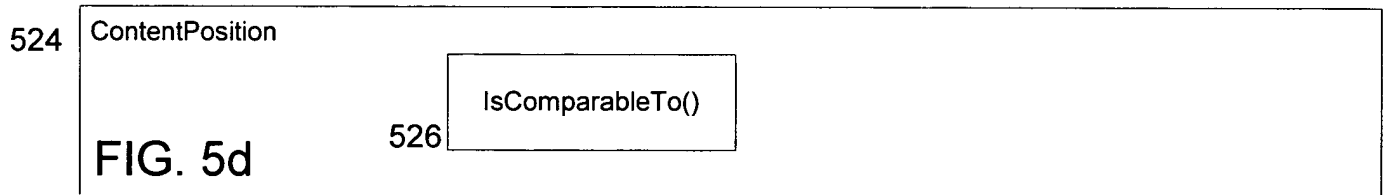
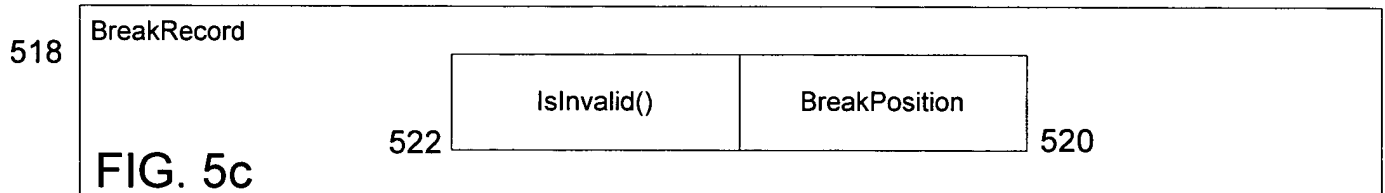
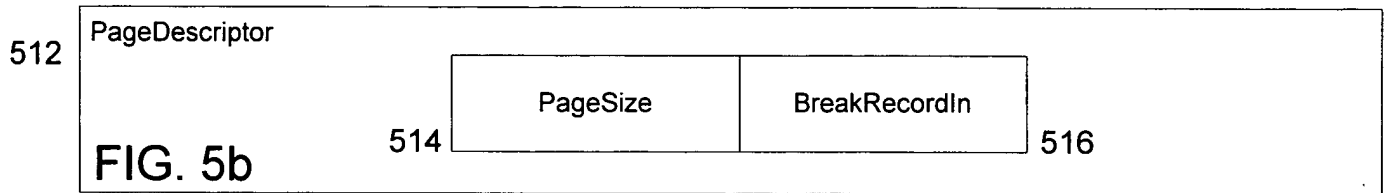
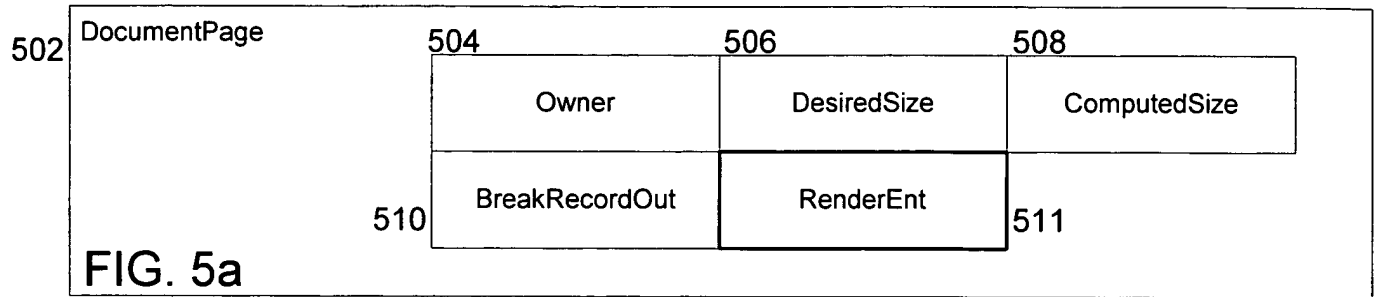


FIG. 4



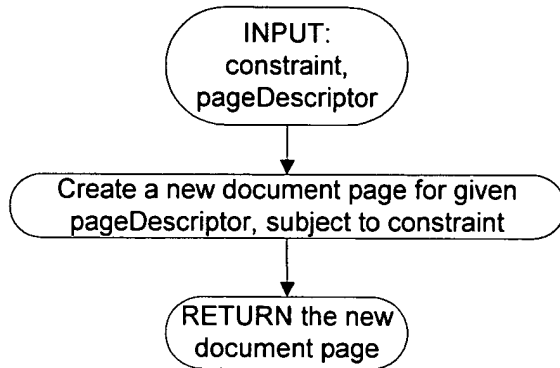


FIG. 6a

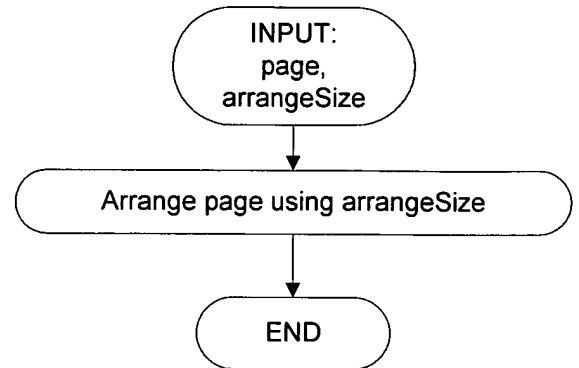


FIG. 6b

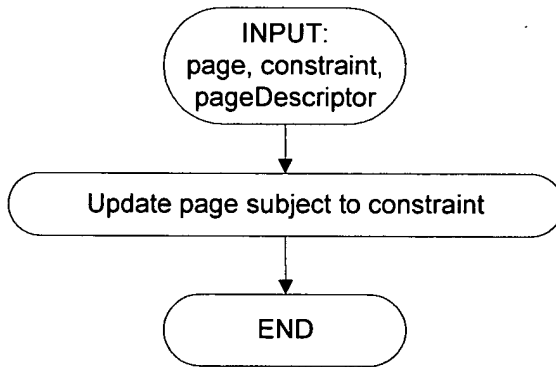


FIG. 6c

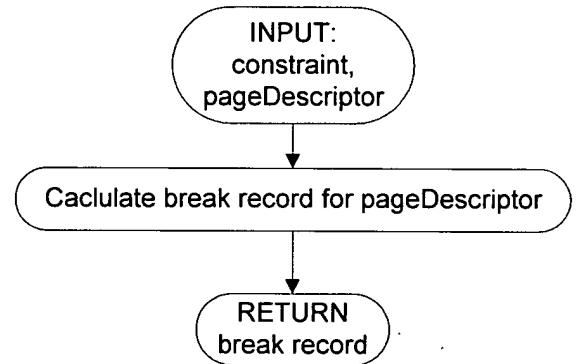


FIG. 6d

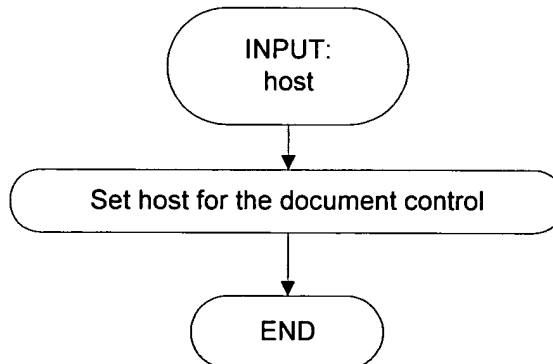


FIG. 6e

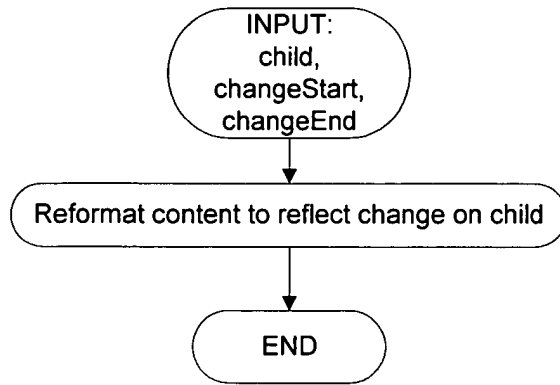


FIG. 7a

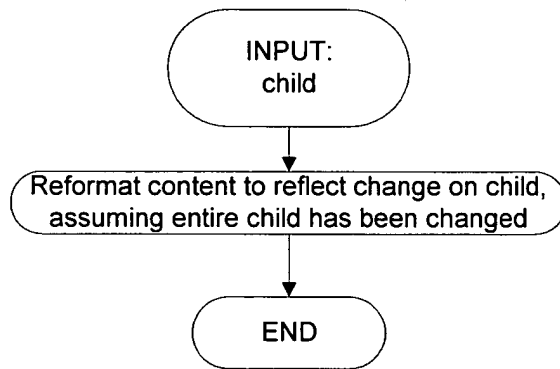


FIG. 7b

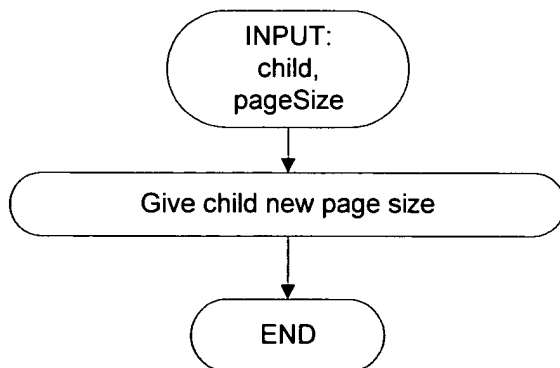


FIG. 7c

```
interface IDocumentContentHost
{
    /// <summary>
    /// Invoked on the host by one of its children (IDocumentFormatter
    /// object), when content is changing. The host needs to reformat
    /// at least part of its content to reflect changes on the child.
    /// </summary>
    /// <param name="child">Child which has been changed.</param>
    /// <param name="changeStart">Start position of a change.</param>
    /// <param name="changeEnd">End position of a change.</param>
    void OnContentChanged(
        IDocumentFormatter child,
        ContentPosition changeStart,
        ContentPosition changeEnd);

    /// <summary>
    /// Invoked on the host by one of its children (IDocumentFormatter
    /// object) when content is changing. The host needs to reformat
    /// at least part of its content to reflect changes on the child.
    /// Assume that entire child has been changed.
    /// </summary>
    /// <param name="child">Child which has been changed.</param>
    void OnContentChanged(
        IDocumentFormatter child);

    /// <summary>
    /// Invoked on the host by one of its children (IDocumentFormatter
    /// object) when content has explicit page size requirement, which
    /// is not matching the current page size (in PageDescriptor).
    /// </summary>
    /// <param name="child">Child which requires new page size.</param>
    /// <param name="pageSize">Page size.</param>
    void OnPageSizeMismatch(
        IDocumentFormatter child,
        Size pageSize);
}
```

FIG. 8


```
interface IDocumentFormatter
{
    /// <summary>
    /// Creates and formats a new document page.
    /// Throws exception if host is not set.
    /// </summary>
    /// <param name="constraint">Size constraint for the page.</param>
    /// <param name="pageDescriptor">Page descriptor; always required.
    /// For bottomless scenarios height is always =  $\infty$ .</param>
    /// <returns>An object representing formatted page.</returns>
    DocumentPage MeasurePage(
        Size constraint,
        PageDescriptor pageDescriptor);

    /// <summary>
    /// Update existing document page.
    /// Throws exception if host is not set.
    /// </summary>
    /// <param name="page">DocumentPage to update.</param>
    /// <param name="constraint">Size constraint for the page.</param>
    /// <param name="pageDescriptor">PageDescriptor; always required.
    /// For bottomless scenarios height is always =  $\infty$ .</param>
    void UpdatePage(
        DocumentPage page,
        Size constraint,
        PageDescriptor pageDescriptor);

    /// <summary>
    /// Arranges existing document page.
    /// Throws exception if host is not set.
    /// </summary>
    /// <param name="page">DocumentPage to arrange.</param>
    /// <param name="arrangeSize">Desired size of the page.</param>
    void ArrangePage(
        DocumentPage page,
        Size arrangeSize);

    /// <summary>
    /// Calculates only break record for the document page.
    /// Throws exception if host is not set.
    /// </summary>
    /// <param name="constraint">Size constraint for the page.</param>
    /// <param name="pageDescriptor">Page descriptor.</param>
    /// <returns>Page break record.</returns>
    BreakRecord CalculateBreakRecord(
        Size constraint,
        PageDescriptor pageDescriptor);

    /// <summary>
    /// Set the host of the document control.
    /// </summary>
    /// <param name="host">Host object, which receives notifications
    /// about content changes.</param>
    void SetDocumentContentHost(
        IDocumentContentHost host);
}
```

FIG. 9

```
abstract class DocumentPage
{
    /// <summary>
    /// Object which DocumentPage is representing.
    /// </summary>
    Object Owner { get; }

    /// <summary>
    /// Desired size of the page.
    /// </summary>
    Size DesiredSize { get; }

    /// <summary>
    /// Computed size of the page.
    /// </summary>
    Size ComputedSize ( get; )

    /// <summary>
    /// BreakRecord indicating break position of the page.
    /// 'null' if the page is bottomless or it is the last page.
    /// </summary>
    BreakRecord BreakRecordOut { get; }

    /// <summary>
    /// RenderableEntity node representing content of the page.
    /// </summary>
    RenderableEntity RenderEnt { get; }

    /* Detailed interface for document related features          */
    /* (footnotes, pagenotes, etc.) to be provided                */
}
*/
*/
```

FIG. 10a

```
sealed class PageDescriptor
{
    PageDescriptor(
        Size pageSize,
        BreakRecord breakRecordIn);

    /// <summary>
    /// Size of the document page (top-level page).
    /// </summary>
    Size PageSize { get; }

    /// <summary>
    /// BreakRecord of the previous page. Necessary to continue
    /// page formatting.
    /// </summary>
    BreakRecord BreakRecordIn { get; }
}
```

FIG. 10b

```
abstract class BreakRecord : IComparable
{
    /// <summary>
    /// Determines if the break record is invalidated by specified
    /// dirty text range (DTR).
    /// </summary>
    /// <param name="changeStart">Start position of a change.</param>
    /// <param name="changeEnd">End position of a change.</param>
    /// <returns>'true' if is invalid, 'false' otherwise.</returns>
    bool IsInvalid(
        ContentPosition changeStart,
        ContentPosition changeEnd);

    /// <summary>
    /// Content break position. If content breaks in more than one
    /// place, this position points to further most break position.
    /// </summary>
    ContentPosition BreakPosition { get; }
}
```

FIG. 11a

```
abstract class ContentPosition : IComparable
{
    /// <summary>
    /// Return true if this instance can be compared to the passed
    /// position.
    /// </summary>
    public abstract bool IsComparableTo(
        ContentPosition position);
}
```

FIG. 11b

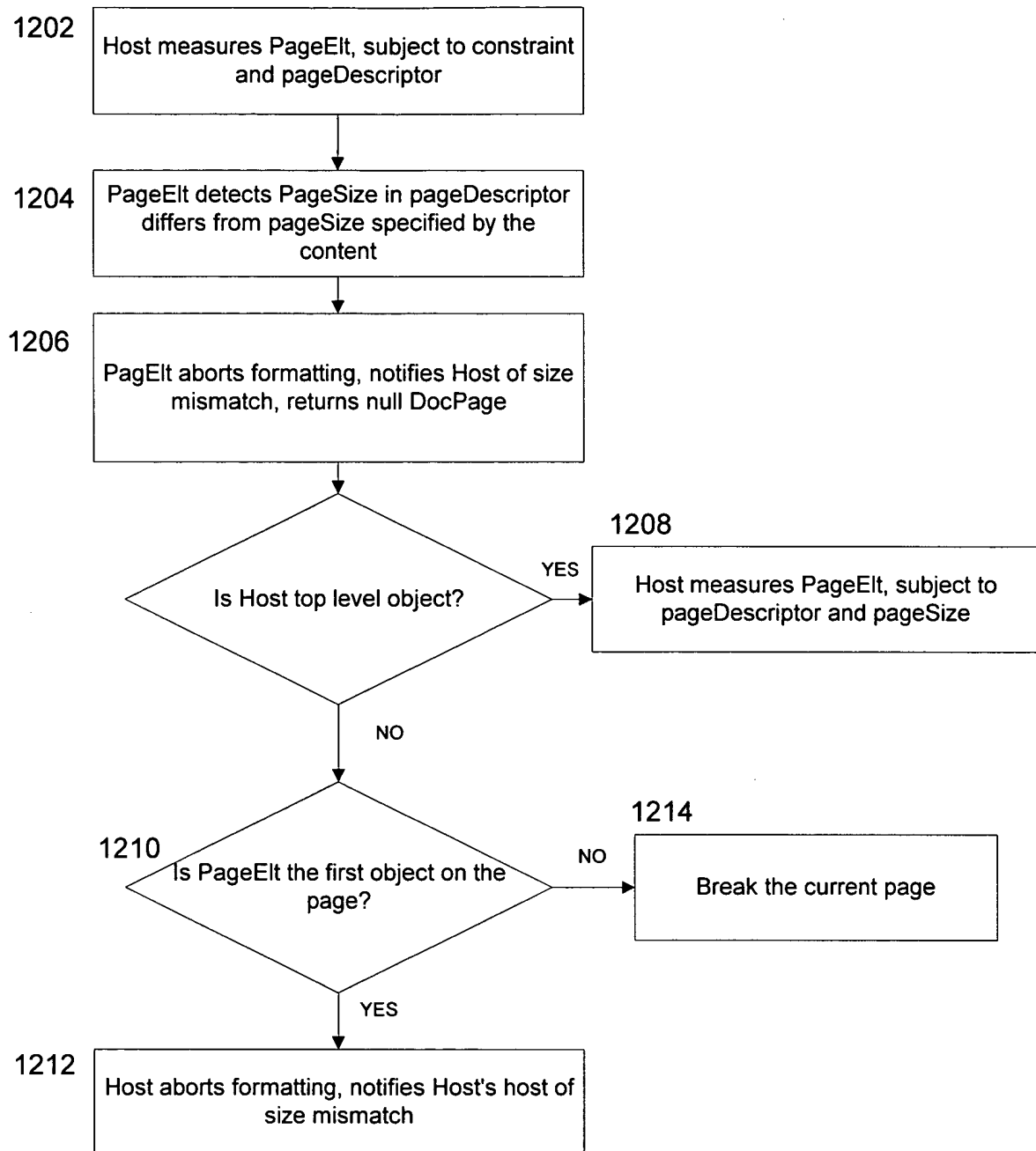


FIG. 12

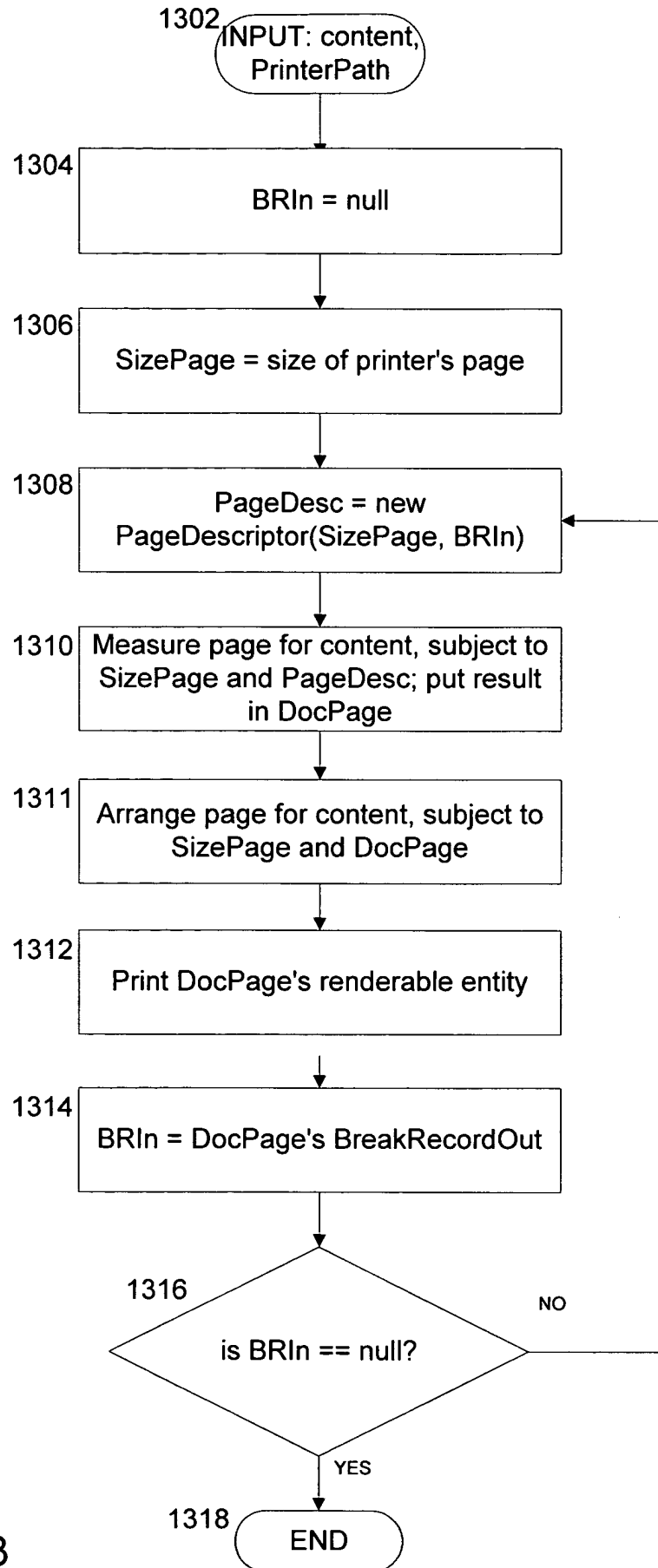


FIG. 13

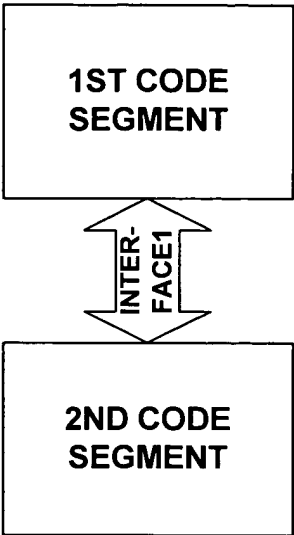


FIGURE A1

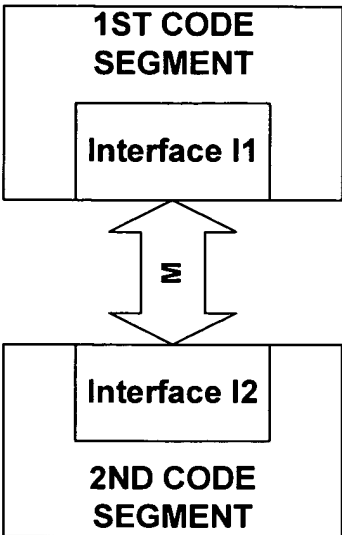


FIGURE A2

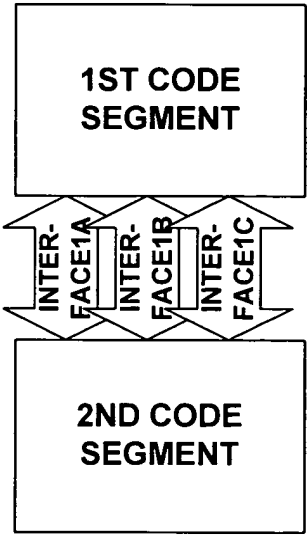


FIGURE B1

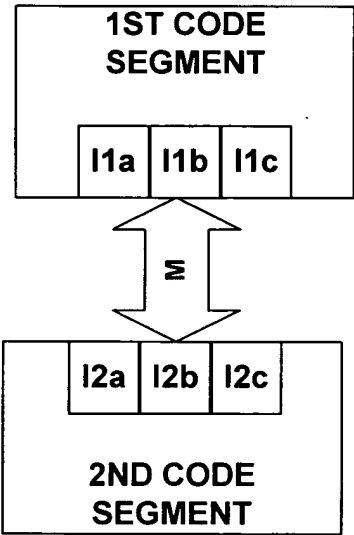


FIGURE B2

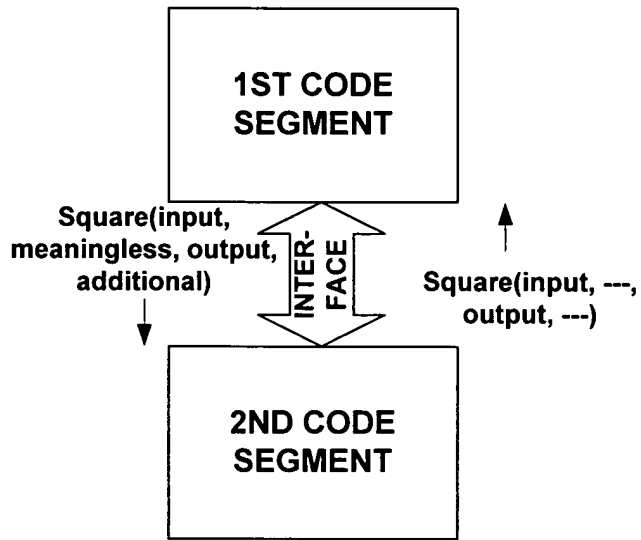


FIGURE C1

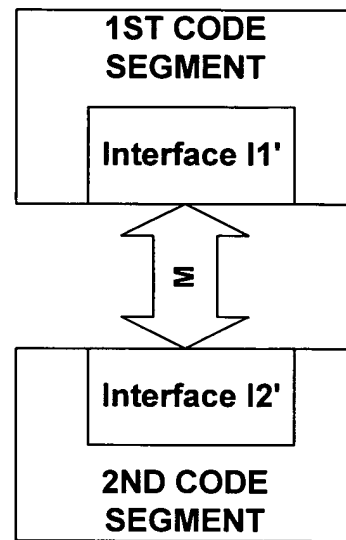


FIGURE C2

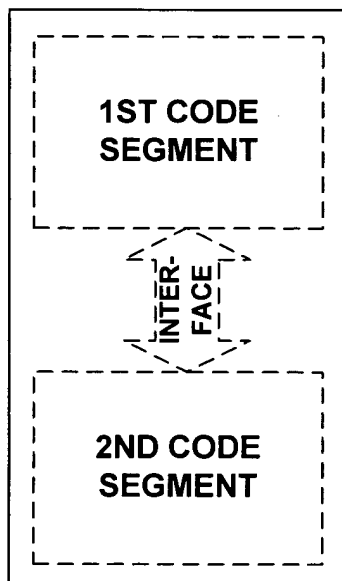


FIGURE D1

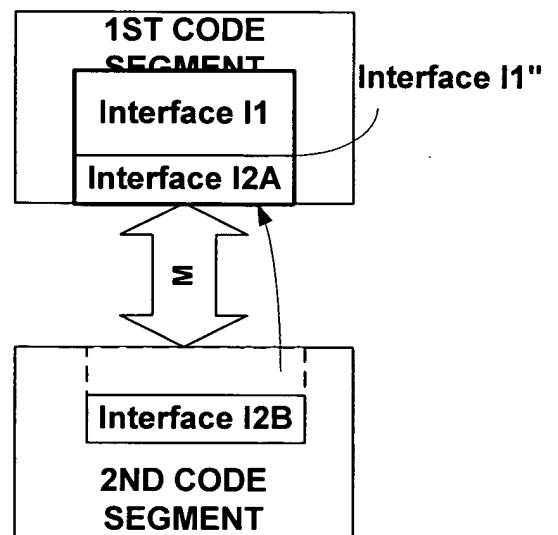


FIGURE D2

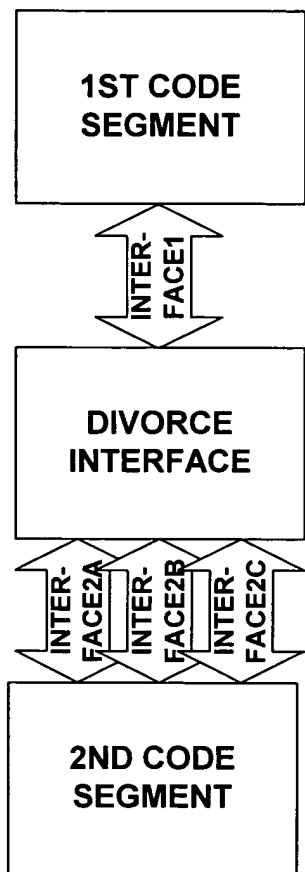


FIGURE E1

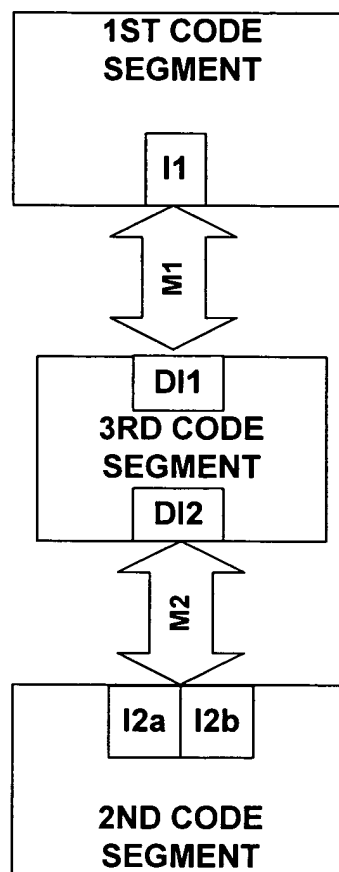


FIGURE E2

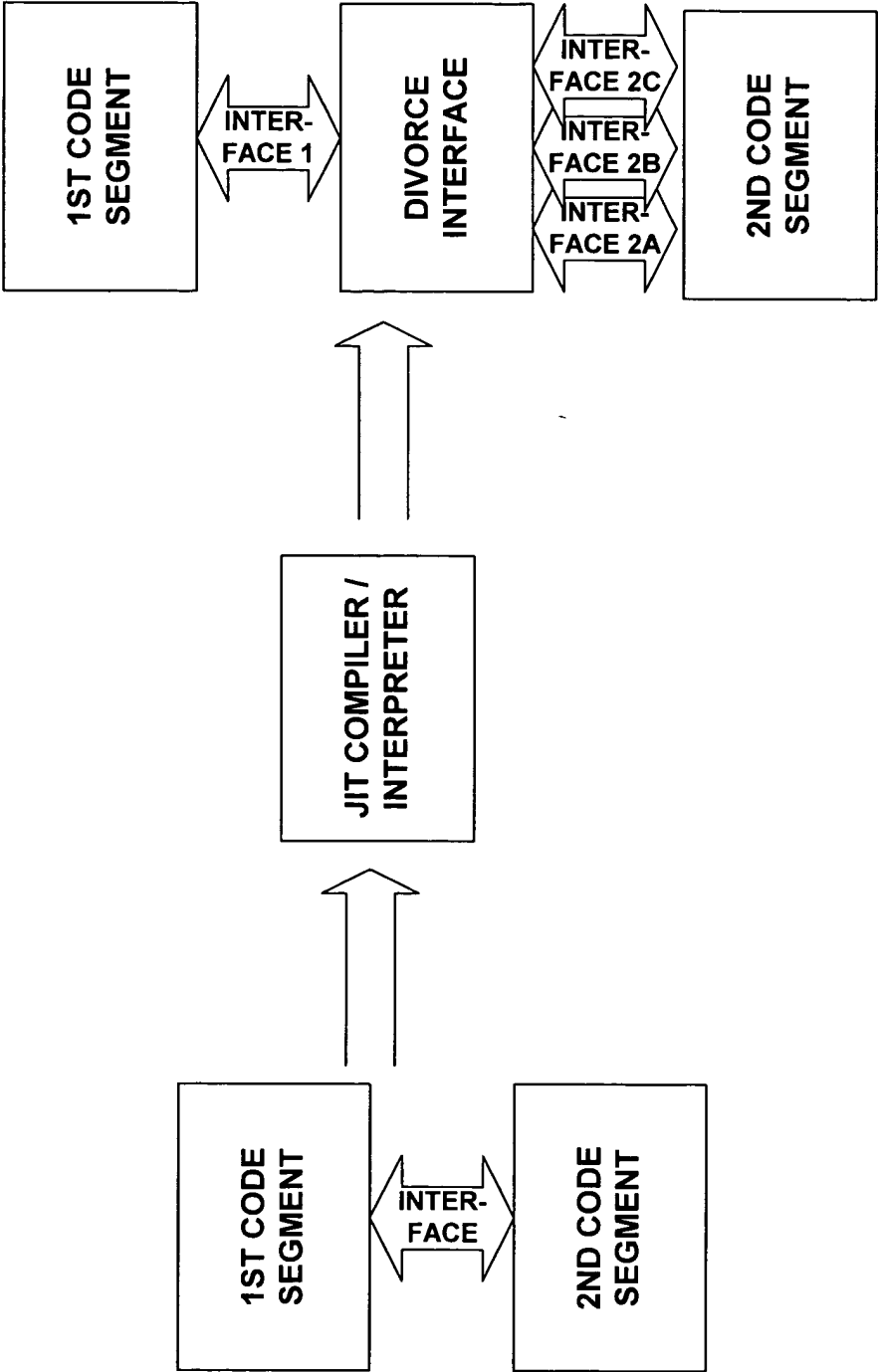


FIGURE F1

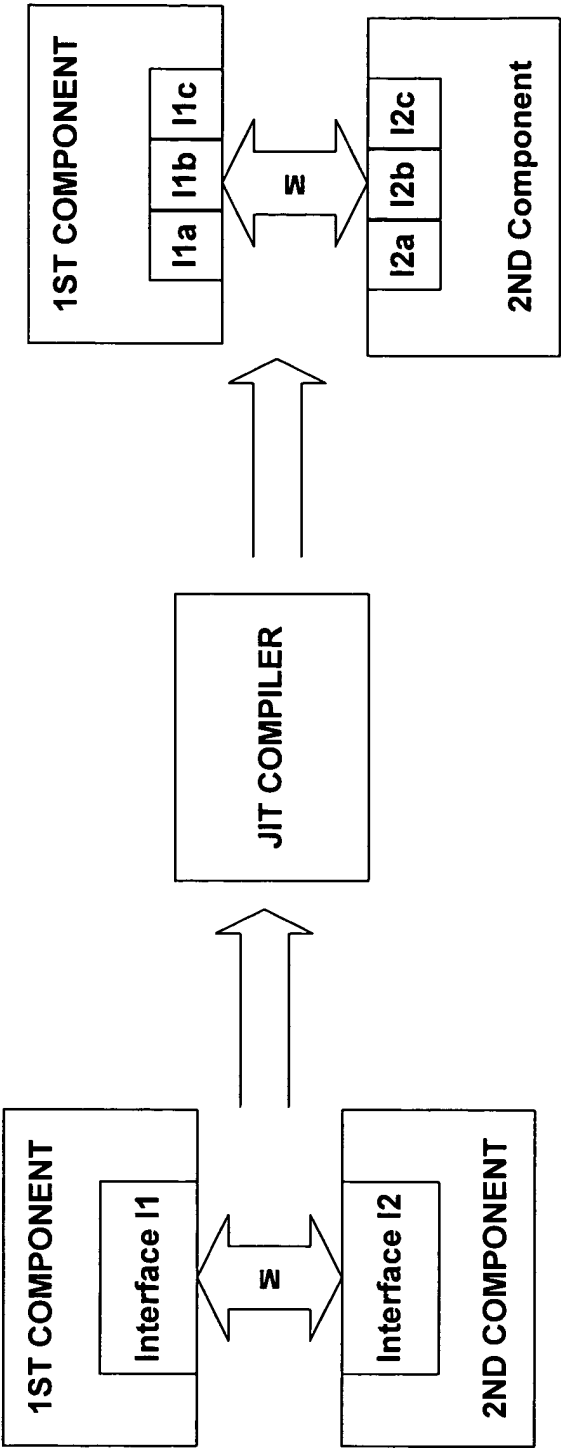


FIGURE F2